REAL-TIME 2–3 PULL-DOWN ELIMINATION APPLYING MOTION ESTIMATION/COMPENSATION IN A PROGRAMMABLE DEVICE

R. J. Schutten and G. de Haan
Philips Research Laboratories, Eindhoven, The Netherlands

ABSTRACT

A software package realizes real-time video processing on a commercially available programmable device[1]. The software implements a motion estimator and a picture rate convertor to provide judder-free display of movie material broadcast in 2–3 pull-down mode. A new object-based true-motion estimation algorithm efficiently uses the VLIW core of the processor. It permits quasi-simultaneous motion estimation/segmentation for a fixed maximum number of objects.

1 INTRODUCTION

Picture sequences are generated at various picture rates: movie material at 24, 25, and 30 Hz, and video usually at 50 and 60 Hz. Simple picture-rate convertors repeat pictures until the next one arrives, which results in blur and/or judder when motion occurs. Knowing the motion allows these effects to be eliminated.

However, Motion Estimation (ME) has only recently reached the quality and price levels needed for consumer equipment [1]. This paper shows new progress, introducing ME and MC (motion compensation) for eliminating judder from 60 Hz TV movies using a software package that runs real time on a programmable device [2]. De-interlacing and movie detection on the VLIW core of the processor has already been reported [3]. The new concept is an extension of that work for high-end TVs and broadcast PCs.

The software package implements motion-compensated picture interpolation with order statistical filtering [4] to guarantee robustness in the event of vector errors. This concept has proven useful before [1]. The ME part of the design, however, is completely new. Rather than estimating vectors for *blocks*, we applied an *object-based* ME method. The software detects video material that originates from film, and automatically adapts to it. The software allows conversion to progressive and interlaced TV formats, and to VGA for PCs.

1. The Philips TriMedia processor is commercially available as the TM1000.

Section 2 of this paper introduces the innovative object-based motion estimation algorithm, Section 3 discusses film-mode recognition, and Section 4 the up-conversion algorithm. Section 5 covers the software implementation, performance and use of resources. Our conclusions are given in Section 6.

2 MOTION ESTIMATION

Motion vectors are used in a wide range of applications such as coding, noise reduction, and scan rate conversion. Some of these applications, particularly frame rate conversion, require the *true-motion* of objects to be estimated [5, 6]. Other applications, for example interlaced-to-sequential scan conversion, demand a high accuracy of the motion vectors to achieve a low amplitude of remaining alias [7, 8]. There is a final category of applications, such as consumer ones, where the cost of the motion estimator is of crucial importance [9, 10].

Several algorithms have been proposed to achieve true-motion estimation [5, 6, 10-13], and algorithms have also been proposed to realize motion estimation at a low complexity [9-11, 14-16]. In addition to the pel-recursive algorithms that usually allow sub-pixel accuracy [17, 18], a number of block-matching algorithms have been reported that yield highly accurate motion vectors [5, 19, 20]. Some years ago, a recursive type of block-matcher was proposed that combines the true-motion estimation required for frame rate conversion with the low complexity constraint needed for consumer applications [11]. A commercial implementation has improved the motion portrayal of film material shown on television [1].

The new motion estimator described in this paper is suitable for scan-rate conversion, with even lower computational complexity than before [11]. The aim was to make the reduction so significant that the algorithm would run in real time on the VLIW core of the processor.

A first reduction is achieved by designing an *object-based* motion estimator instead of a *block-based* motion estimator. We expect fewer objects than blocks in realistic images, which implies that fewer motion parameters have

Manuscript received June 17, 1998

to be estimated. We have assumed that the motion of each object can be described with a parametric model.

A second reduction is achieved by reducing the resolution of the images on which the motion is estimated (sub-sampling).

The new object-based motion estimator can distinguish a maximum of three different objects. Here, "objects" mean image section(s) that can be described using the same motion model, and do not necessarily correspond to a single physical object in the scene.

In the following subsections, we shall describe the motion model, the estimation of the model parameters, the cost function, and the segmentation of the image.

2.1 Motion model

To keep complexity low, the motion of an object l is described by a simple translational model[1],

$$\vec{D}(\vec{x}, l, n) = \begin{pmatrix} s_x(l,n) \\ s_y(l,n) \end{pmatrix} \tag{1}$$

using $\vec{D}(\vec{x}, l, n)$ for the displacement vector of object l at location $\vec{x} = (x, y)^T$ in the image with index n.

2.2 Parameter estimation

Given a motion model, the next problem is to estimate its parameters to yield a useful description for an object in the image. As it is vital for almost every sequence to deal correctly with stationary image sections, the algorithm starts with an 'object 0', for which motion is described by the zero vector (no estimation effort is required for this). The parameter vectors of additional objects l, $l > 0$ are then estimated separately and in parallel, as shown in Figure 1, by their respective parameter estimators (PE_l). Each PE_l has a basic principle very similar to that of the 3D recursive search block matcher of [11]. A previously estimated parameter vector is updated, after which the best parameter vector is selected according to a cost function.

Considering the two-parameter model of eq. (1), the parameters of object l, $l > 0$, are regarded as a parameter vector P_l:

$$\vec{P}_l(n) = \begin{pmatrix} s_x(l,n) \\ s_y(l,n) \end{pmatrix} \tag{2}$$

and we define our task as being to select $\vec{P}_l(n)$ from a number of candidate parameter vectors $\overrightarrow{CP}_l(n)$ as the one that has the minimum value of a cost function, to which we shall return later.

1. More complex parametric motion models have been proposed [21] and can indeed be used in combination with the proposed algorithm, but will be disregarded here.

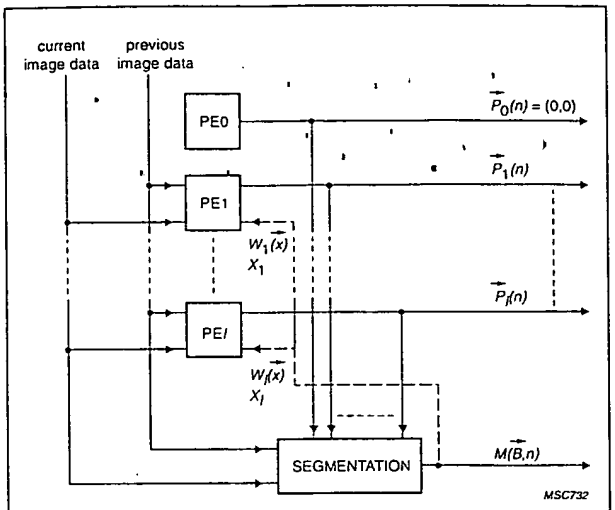

Fig. 1 Block diagram of multiple parameter estimators and segmentation.

The candidates are generated in a very similar manner to the strategy exploited in [11]: we take a prediction vector, add at least one update vector, and select the best candidate parameter vector according to cost function eq. (5). Candidate parameter set $CPS_l(n)$ contains three candidates $\overrightarrow{CP}_l(n)$ according to:

$$CPS_l(n) = \{\overrightarrow{CP}_l(n) \mid \overrightarrow{CP}_l(n) = \vec{P}_l(n-1) + m\overrightarrow{UP}_l(n), \; \overrightarrow{UP}_l(n) \in S_{\overrightarrow{UP}_l}(n), \; m = -1, 0, 1\} \tag{3}$$

with update parameter $\overrightarrow{UP}_l(n)$ selected from update parameter set $UPS_l(n)$:

$$UPS_l(n) = \{\begin{pmatrix} i \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ i \end{pmatrix}\}, \quad (i = 1, 2, 4, 8, 16) \tag{4}$$

2.3 The cost function

Given the motion model and some candidate parameter sets, we need to select the best candidate for a given object according to a cost function result. The cost function can be a sum of absolute differences between motion compensated pixels from neighbouring images, with vectors generated by the (candidate) motion model. However, we need to know the area to which the motion model is to be assigned.

We now assemble the recent history set $RH(n)$ as:

$$RH(n) = \{\max(n), \max(n-1), \max(n-2), \max(n-3), \max(n-4), \max(n-5), \max(n-6)\} \quad (9)$$

which with adaptive thresholding is converted into a binary movie detection set $MD(n)$ for 2-2 pull-down, that will give something like:

$$MD(n) = \{0, 1, 0, 1, 0, 1, 0\} \quad (10)$$

for 2-3 pull-down something like:

$$MD(n) = \{0, 1, 0, 0, 1, 0, 1\} \quad (11)$$

and for video something like:

$$MD(n) = \{1, 1, 1, 1, 1, 1, 1\} \quad (12)$$

Comparing the actual set with a limited number of known patterns yields information on movie type and phase. For scene cuts, the detector output is unreliable, and motion compensation should be switched off.

4 THE UP-CONVERSION ALGORITHM

The software supports conversion to progressive TV formats and to (progressive) VGA for PCs. The input material therefore needs de-interlacing. All supported input picture rates are also converted to the correct output picture rate, so temporal interpolation is also required.

Material originating from film needs temporal interpolation. The de-interlacing in this case is a field merging, which requires no operations other than correct memory addressing.

Material originating from video cameras needs no temporal interpolation as the output picture rate matches the input picture rate. However, field merging is not suitable as a de-interlacing method, since with motion both fields show objects at different positions. Therefore, a vertical-temporal median filter is implemented for de-interlacing.

Depending on the film mode, the software operates either in a temporal interpolation mode for film or in a de-interlacing mode for video.

Note that the resources for motion estimation are necessary all the time, as without motion vectors it is impossible to reliably discriminate between video and film.

4.1 Temporal interpolation

An MC temporal interpolation algorithm with order statistical filtering guarantees robustness in the event of vector errors. This concept has proved useful before, in [1].

A straightforward motion compensated average would yield an intermediate picture according to:

$$F_i(\vec{x}, n) = \frac{1}{2}\{F(\vec{x} - \alpha\vec{D}(\vec{x}, n), n-1) + F(\vec{x} + (1-\alpha)\vec{D}(\vec{x}, n), n)\} \quad (13)$$

where α determines the temporal position of the interpolated image. We implement an order statistical filter to make the up-conversion more robust when coping with erroneous motion vectors. A very basic version was described in [4]. This version uses:

$$F_i(\vec{x}, n) = \text{med}\{F(\vec{x} - \alpha\vec{D}(\vec{x}, n), n-1), Av, F(\vec{x} + (1-\alpha)\vec{D}(\vec{x}, n), n)\} \quad (14)$$

with

$$Av = \frac{1}{2}\{F(\vec{x}, n) + F(\vec{x}, n-1)\} \quad (15)$$

and:

$$\text{med}(a, b, c) = \begin{cases} a & , (b \le a \le c \vee c \le a \le b) \\ b & , (a \le b \le c \vee c \le b \le a) \\ c & , (\text{otherwise}) \end{cases} \quad (16)$$

4.2 De-interlacing

A vertical-temporal median filter [23] is used for de-interlacing video camera signals. As the temporal interpolation is required for movie material only, no additional resources are claimed for video. The processing power required for robust up-conversion implementing equation (14) for movie material, is consumed by interpolating intermediate lines for the current field n using equation (17) in case of video material.

$$F_i(\vec{x}, n) = \text{med}\{F(\vec{x}, n-1), F\left(\vec{x} + \begin{pmatrix}0\\1\end{pmatrix}, n\right), F\left(\vec{x} - \begin{pmatrix}0\\1\end{pmatrix}, n\right)\} \quad (17)$$

5 IMPLEMENTATION ON A PROGRAMMABLE DEVICE

Figure 4 shows the block diagram of the processor on which the application runs. Figure 5 shows the sections of the algorithm that are running on the VLIW core of the processor. Table 1 shows some of the options that are implemented in the software.

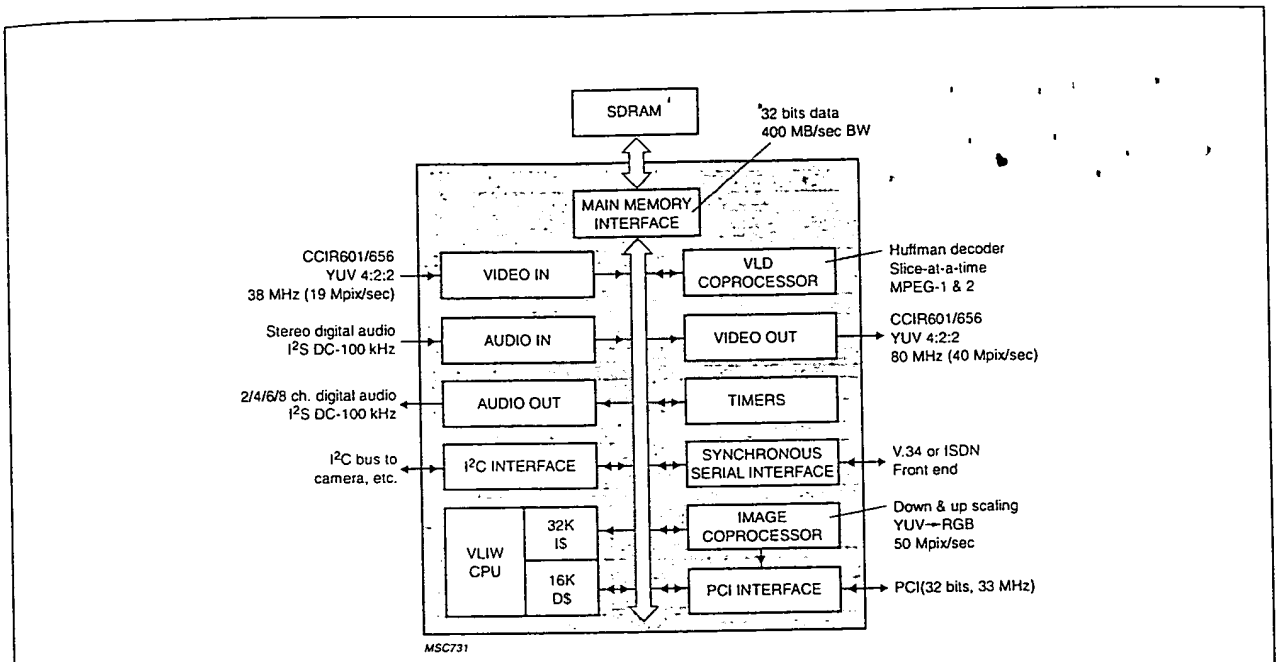

Fig.4 Block diagram of the processor.

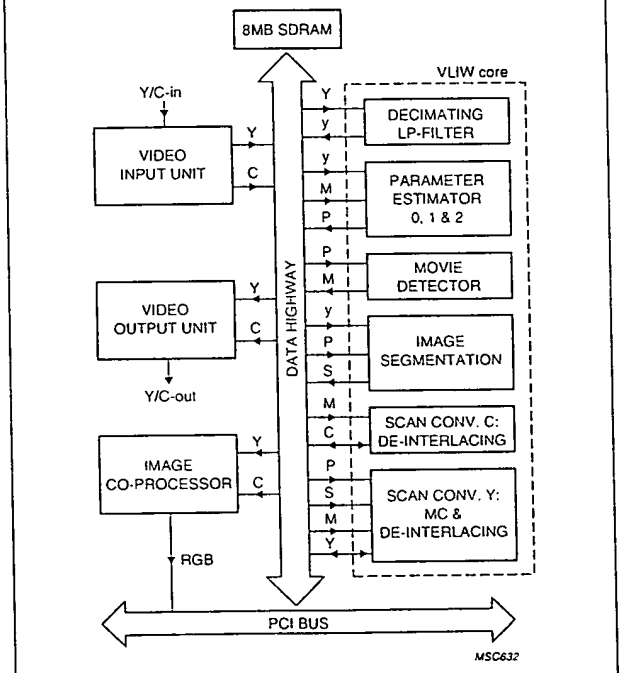

Fig.5 Block diagram of the processing.

Table 1 Conversion characteristics of the software.

Picture formats:			
Input	**Picture rate (pic./sec)**	**Output**	
525 lines/2:1	24, 30 (movie); 60 (video)	60 Hz (direct)	720×480/2:1 (CCIR601)
			720×480/1:1
		60 Hz (PCI)	640×480/1:1 (VGA)
625 lines/2:1	25 (movie); 50 (video)	50 Hz (direct)	720×576/2:1 (CCIR601)
			720×576/1:1
Velocity range (pixels/picture-period):			
Horizontal:		+/– 32	
Vertical:		+/– 16	
De-interlacing algorithm:			
Movie:		Field-merging	
Video:		VT-median filtering	
Motion compensation algorithm:			
Movie:		Order statistical filtering	
Video:		None (not required)	

The next two subsections discuss the performance of the new *object-based* algorithm (which is somewhat worse than our reference *block-based* algorithm) and the resource usage (which is better).

5.1 Performance

The innovation in this system is the *object-based* motion estimator. It scores somewhat worse on the Modified Mean

These two issues, segmentation and motion estimation, are inter-dependent. To estimate the motion in an object correctly, the area of the object should be known and vice versa.

We circumvented the chicken-and-egg problem by introducing a hierarchy in the parameter estimators and by using segmentation masks calculated for the previous image. Each parameter estimator PE_l will calculate its cost function on the set of locations defined in set X_l (called 'points of interest'). Different locations can be assigned within set X_l using a different weighting factor $W_l(\vec{x})$ dependent on location $\vec{x}$.

The hierarchy in the estimators is achieved by:

- Selecting a set of points of interest $\vec{x}$ in X_l as found in the *previous* image, by excluding locations sufficiently covered by higher ranked objects in the previous image. Each estimator, apart from the highest in the hierarchy (the zero estimator), minimizes a cost function calculated for objects in which all higher level estimators were unsuccessful in the *previous* image. The set of points of interest $\vec{x}$ in X_l is filled with the positions of $\vec{x}$ where the error function of all higher ranked objects exceeds the average block match error with a fixed factor. A correct selection of X_l is necessary to prevent the current estimator from estimating motion that is already covered by higher-ranked parameter estimators.
- Reducing the effect of locations within X_l that are potentially better covered by objects ranked lower in the hierarchy: assigning higher weights $W_l(\vec{x})$ to the pixels assigned to object l in the *previous* segmentation. The location dependent weighting factor $W_l(\vec{x})$ is determined by the segmentation mask $M(B, n-1)$ (see Section 2.4) found in the *previous* image. Positions $\vec{x}$ that belong to the current object l according to the segmentation mask will have a weighting factor greater than one, where positions belonging to a different object have a weighting factor of one. A correct selection of $W_l(\vec{x})$ is necessary to prevent the current estimator from estimating motion that can be covered by lower-ranked parameter estimators.

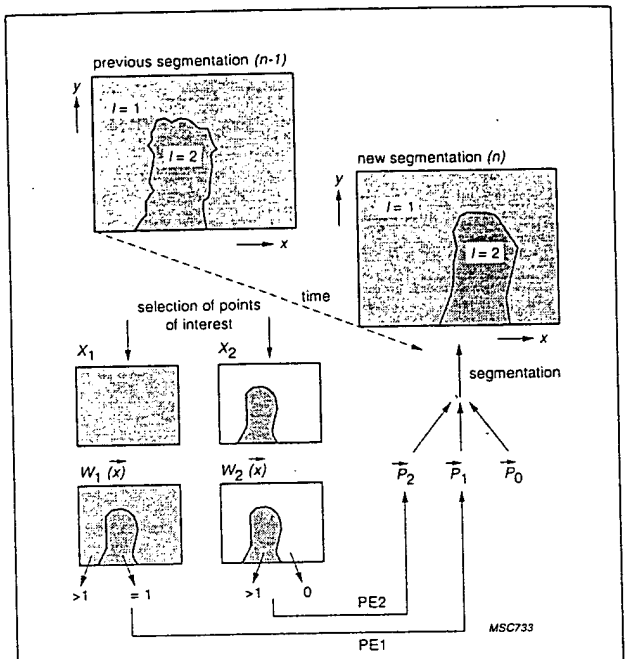

Fig.2 Example of selection of points of interest.

Figure 2 gives an example of how the points of interest are selected in an image with two moving objects. The moving background of the image is object $l = 1$, and there is a smaller object $l = 2$. Set X_1 will be the entire image (everything not sufficiently covered by $l = 0$), and X_2 covers the part of the image that is not sufficiently covered by $l = 0$ and $l = 1$. In the part of the image that was covered by object $l = 1$ in the previous segmentation, $W_1(\vec{x}) > 1$ results, with $W_1(\vec{x}) = 1$ in the section covered by other objects. In the section covered by object $l = 2$, $W_2(\vec{x}) > 1$ results. Since X_2 does not extend beyond the area of object $l = 2$, there is no section where $W_2(\vec{x})$ equals one.

More formally, the cost function is calculated according to:

$$\varepsilon'(\overrightarrow{CP_l}(n)) = \varepsilon(\overrightarrow{CP_l}(n)) + \sum_{\vec{x} \in X_l} W_l(\vec{x}) \cdot \Pi(\overrightarrow{CP_l}(n)) \quad (5)$$

where penalties $\Pi(\overrightarrow{CP_l}(n))$ are added to the match error of individual candidate vectors (parameters sets) to obtain temporal smoothness, and ε is:

$$\varepsilon(\overrightarrow{CP}_l(n)) = \sum_{\vec{x} \in X_l} W_l(\vec{x}) \cdot \left| F_s(\vec{x}, n) - F_s(\vec{x} - \vec{D}(\vec{x}, l, n), n-1) \right| \quad (6)$$

Here $F_s(\vec{x}, n)$ is the luminance value at position $\vec{x}$ in a *sub-sampled* image with index n.

The sub-sampling effectively reduces the required memory bandwidth. Images are sub-sampled with a factor of 4 horizontally and 2 vertically on a field base, generating a sub-sampled image $F_s(n)$ from each original field $F(n)$. To achieve pixel accuracy on the original pixel grid of F, interpolation is required on the sub-sampling grid.

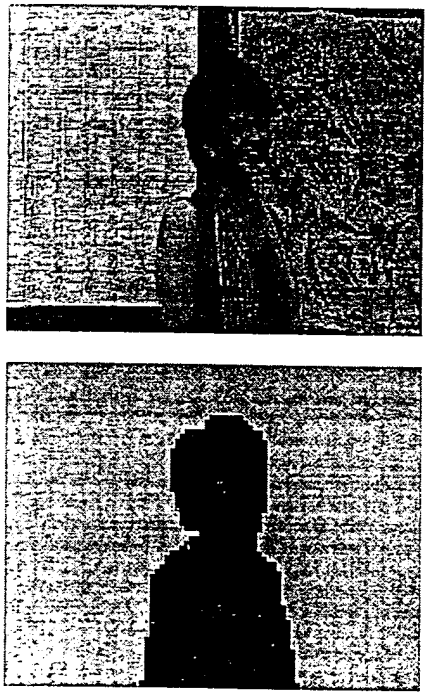

Fig.3 Picture and resulting segmentation.

2.4 Segmentation

Segmentation is the most critical step in the algorithm. Its task is to assign one motion model to each group of pixels. This is basically achieved by assigning the best matching model to each group of pixels (a block B, which is typically as large as 8×8 pixels on frame base).

For each block, a match error is calculated according to:

$$\varepsilon(\vec{B}, l, n) = \sum_{\vec{x} \in B} \left| F_s(\vec{x} + (1-\alpha)\vec{D}(\vec{x}, l, n), n) - F_s(\vec{x} - \alpha\vec{D}(\vec{x}, l, n), n-1) \right| \quad (7)$$

Segmentation mask $M(B, n)$ assigns the object l with the lowest ε to the block B. The temporal position of the segmentation is defined by α, which was set to 1/2 in our experiments.

To save processing power, the segmentation mask M does not need to be calculated for every block B. Instead, the calculated blocks can be sub-sampled in a quincunx pattern, after which the missing positions in the segmentation mask are interpolated (for example by choosing the most frequently occurring object number from a neighbourhood) [11].

Segmentation is more difficult when many objects occur in the image, since the segmentation task will resemble more and more that of a full search block matcher. Extra (smoothing) measures have been added to prevent an output from the motion estimator having inconsistencies similar to those of a full search block matcher.

These measures are:

- *Overlapping blocks*: by taking a larger window when calculating the ε than the size of the block B to which the object is assigned (spatial smoothing).
- *Bonus system*: by reducing the calculated ε of an object using a bonus value if this object was chosen in the segmentation of the previous image or in spatially neighbouring blocks (temporal and spatial smoothing).

Figure 3 gives an example of a segmentation according to the *object-based* motion estimation method, with the original luminance image.

3 FILM-MODE RECOGNITION

The movie detector that we described in [22] recognizes movie and video formats using motion vectors integrated over a field period of the video signal. The current detector recognizes both 2–3 and 2–2 pull-down. Furthermore, since the parameter estimators already describe motion for a large portion of the image, vectors no longer need to be integrated. We found that a reliable movie detector can be realized by analysing the motion in the highest ranked estimator only, disregarding the zero-vector 'estimator'.

Let us define $\max(n)$ as the largest component of parameter vector $P_1(n)$, i.e.

$$\max(n) = max\{s_x(1, n), s_y(1, n)\} \quad (8)$$

Square Error (M^2SE) of the motion vector field than the reference *block-based* motion estimator (Table 2).

Table 2 Relative performance of the motion estimator

Modified Mean Square Error			
Scene	Block-based	Object-based	(Object-based/Block-based) (%)
Bond 1	40	46	115
Bond 2	124	129	104
Girl & Fence	100	120	120
PRL Car	81	108	133
Renata	38	174	458
Subtext	44	98	223
Total	427	675	158

M^2SE is a measurement of the true motion quality of a vector field [11]. The M^2SE is defined as:

$$M^2SE(n) = \frac{1}{PL} \sum_{\vec{x} \in MW} [F(\vec{x}, n) - F_{mc}(\vec{x}, n)]^2 \qquad (18)$$

where $F(\vec{x}, n)$ is a frame of the test sequence on which $F_{mc}(\vec{x}, n)$ is also calculated. PL is the number of pixels in the measurement window MW that corresponds to the entire image, excluding a margin of $M \times N$ at the edges, where M and N define the vector range of the estimator (Table 1). The interpolated picture $F_{mc}(\vec{x}, n)$ is a motion compensated average resulting from shifted pictures at $n-1$ and $n+1$:

$$F_{mc}(\vec{x}, n) = \frac{1}{2}[F(\vec{x} - \vec{D}(\vec{x}, n), n-1) + F(\vec{x} + \vec{D}(\vec{x}, n), n)] \qquad (19)$$

For more background on the M^2SE, see [24]. The average M^2SE over a 20 frame sequence is calculated to improve reliability.

Table 2 shows that scenes with only camera panning (Bond 1) or a few fairly large moving objects (Bond 2, Girl & Fence, PRL Car) are only slightly degraded compared with scenes that can be considered difficult for the object-based algorithm-like scenes with objects having almost the same velocities (Renata) or with small moving objects (Subtext).

Subjectively, the results on the first four sequences (Bond 1 through PRL Car) are perceptually acceptable. The perceptual performance of the last two sequences (Renata and Subtext) is unacceptable compared to the reference *block-based* algorithm. Future research will focus on improving this performance.

5.2 Resource usage

Table 3 shows the CPU load for the various software modules, and the total usage for the various formats. Formats that require more temporal interpolation clearly require more processing power (2–3 vs. 2–2 pull-down movie). As can be seen, the load of the parameter estimation and segmentation is not high compared to that of the scan conversion modules. This is partly achieved by operating the estimator and segmentation modules on down-scaled images as discussed in Section 2. The load of the segmentation depends on the film mode, because the segmentation operates on the real, received picture rate.

Table 3 Resources claimed by the software.

CPU USAGE (REAL-TIME OPERATION)[1]			
Mode	Video (%)	2–3 movie (%)	2–2 movie (%)
Decimation	5	5	5
Parameter estimation	6	5	4
Movie detection	0	0	0
Segmentation	28	11	14
Scan conversion Y	36	40	31
Scan conversion C	9	0	0
Total	84	61	54

1. Image size: CCIR601, processor: TM1000 @ 132 MHz.

Compared with the *block-based* algorithm described in [1, 11], the *object-based* approach saves roughly a factor of five in calculating the match errors. This can be concluded from the following global calculations.

The *block-based* approach calculates the match error for eight candidates on full resolution images with a factor of four sub-sampling on the match error calculation:

$$8 \text{ candidates} \times \frac{720 \times 480}{4} \div (720 \times 480) = 2 \text{ match errors/pixel.}$$

The *object-based* approach calculates the match error for a maximum of three candidates on a factor eight down-scaled image with all pixels included in match error calculation:

$$3 \text{ candidates} \times \frac{720 \times 480}{8} \div (720 \times 480) \approx 0.4 \text{ match errors/pixel.}$$

This gives a reasonable indication, because the segmentation covers more than 70 % of the resources that are occupied by the motion estimator (see Table 3 under the video column).

6 CONCLUSION

Recent progress in the fields of ME, MC and CPU-architectures has resulted in a software package running in real time on a commercially available processor, realizing judder-free motion of film material on TV and PC displays. Automatic adaptation of the processing to movie and video is included in the software, which supports many input and output scanning formats. This concept has proven useful before [1]. The ME part of the design, however, is completely different. Rather than estimating vectors for *blocks*, we used a newly designed *object-based* ME method.

The new *object-based* motion estimator scores on average 60 % worse based on Modified Mean Square Error (M^2SE) than a *block-based* algorithm. There is, however, a clear distinction between scenes that are on average only 20 % worse and scenes that are more than double the M^2SE. The scenes with only 20 % worse M^2SE are perceptually acceptable, while the scenes with more than double the M^2SE are perceptually unacceptable. Future research is required to improve on this performance.

The new *object-based* motion estimator saves roughly a factor five in resource usage compared to the *block-based* algorithm based on match error calculations.

A maximum of three different objects can be distinguished in the image. In this context, "objects" means image sections(s) that can be described by the same motion model, and do not necessarily correspond to a single physical object in the scene.

The software also includes a movie detector to allow automatic adaptation to film material. Similar to the movie detector of [22], it recognizes movie and video formats using motion vectors integrated over a field period of the video signal. The detector recognizes 2–3 pull-down as well as the 2–2 pull-down detector of [22].

The software supports conversion to progressive and interlaced TV formats, and to VGA for PCs. De-interlacing is therefore required, for which we applied field merging for movie material, and a vertical-temporal median filter [23] for video camera signals.

ACKNOWLEDGEMENTS

The authors wish to thank Bram Riemens and Gerben Hekstra for their contribution to the implementation and the software optimization of the algorithm, and their interaction with the authors for the design of the algorithm.

REFERENCES

[1] G. de Haan, J. Kettenis, and B. Deloore, "IC for motion compensated 100 Hz TV with a smooth-motion movie-mode", *IEEE, Transactions on Consumer Electronics*, vol. 42, pp. 165-174, May 1996.

[2] S. Rathnam and G. Slavenburg, "An architectural overview of the programmable multimedia processor TM 1", in *Proc. Compcon*, pp. 319-326, IEEE CS Press, 1996.

[3] A. Riemens, R. Schutten, and K. Vissers, "High-speed video de-interlacing with a programmable trimedia VLIW core", in *Proc. of the ICSPAT'97*, (San Diego), pp. 1375-1380, September 1997.

[4] G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "Graceful degradation in motion-compensated field-rate conversion", in *Proceedings of the International Workshop on HDTV*, (Ottawa, Canada), pp. 249-256, 1993.

[5] G. Thomas, "Television motion measurement for DATV and other applications", *BBC Research Report*, no. BBC RD 1987/11, 1987.

[6] R. Thoma and M. Bierling, "Motion compensating interpolation considering covered and uncovered background", *Signal Processing: Image Communications 1*, pp. 191-212, 1989.

[7] Kwon, Seo, Kim, and Kim, "A motion-adaptive de-interlacing method", *IEEE Transactions on Consumer Electronics*, vol. 38, pp. 145-150, Aug. 1992.

[8] F. Wang, D. Anastassiou, and A. Netravali, "Time-recursive de-interlacing for IDTV and pyramid coding", *Signal Processing: Image Communications 2*, pp. 365-374, 1990.

[9] G. de Haan and H. Huijgen, "New algorithm for motion estimation", in Chiariglione [25], pp. 109-116.

[10] G. de Haan and H. Huijgen, "Motion estimation for TV picture enhancement", in *Signal Processing of HDTV III* (H. Yasuda and L. Chiariglione, eds.), pp. 241-248, Elseviers Science Publishers B.V., 1992.

[11] G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "True motion estimation with 3-D recursive search block-matching", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 368-388, Oct. 1993.

[12] J. Konrad and E. Dubois, "A comparison of stochastic and deterministic solution methods in bayesian estimation of 2-D motion", *Image and Vision Computing*, vol. 8, pp. 304-317, Nov. 1990.

[13] T. Reuter, "A modified block-matching algorithm with vector reliability checking and adaptive smoothing", in *Third International Conference on Image Processing and its Applications*, (England), University of Warwick, Jul. 1989.

[14] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Transactions on Communications, COM-29*, no. 12, 1981.

[15] T. Koga, K. Iinuma, A. Hirano, Y. Iilima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing", in *IEEE, Proceedings of the NTC 81, G5.3.1.*, (New Orleans), 1981.

[16] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation", *IEEE Transactions on Communication*, no. 8, pp. 888-896, 1985.

[17] H. Musmann, P. Pirsch, and J. Grallert, "Advances in picture coding", *Proceedings of the IEEE*, vol. 73, pp. 523-548, Apr. 1985.

[18] A. Netravali and J. Robbins, "Motion-compensated television coding", *Bell Systems Technical Journal*, no. 3, pp. 629-668, 1979.

[19] K. Hildenbrand and J. Mayer, "Method to determine motion vectors for blocks in animage source-sequence", Patent, no. DE 40 23 449 Cl, 23-01-92. (in German).

[20] M. Ziegler, "Hierarchical motion estimation using the phase correlation method in 140 Mbit/s HDTV-coding", in Chiariglione [25], pp. 131- 137.

[21] A. Tekalp, "Digital Video Processing", *Prentice Hall Signal Processing Series*, ISBN o-13-190075-7, pp. 200-203, 1995.

[22] G. de Haan, P. Biezen, and O. Ojo, "An evolutionary architecture for motion-compensated 100 Hz television", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 207-217, June 1995.

[23] T. Doyle and M. Looymans, "Median filtering of television images", in *Proc. of the ICCE*, (Chicago), pp. 186-187, June 1986.

[24] G. de Haan and P. Biezen, "Sub-pixel motion estimation with 3-D recursive search block-matching", *Signal Processing: Image Communications 6*, pp. 229-239, 1994.

[25] L. Chiariglione, ed., *Signal Processing of HDTV II*, Elseviers Science Publishers B.V., 1990.

BIOGRAPHY

Robert Jan Schutten was born in Groningen, The Netherlands, on January 2, 1972. He received his Master's degree in Electrical Engineering from Delft University of Technology in 1994. In 1995 he joined the Philips Research Laboratories in Eindhoven, where he works as a research scientist in the Television Systems group. His areas of interest are algorithms for image enhancement, motion estimation and scan-rate conversion with a special focus on low-cost software implementations.

Gerard de Haan was born in Leeuwarden, The Netherlands, on April 4, 1956. He received his B.Sc., M.Sc., and Ph.D. from Delft University of Technology in 1977, 1979 and 1992 respectively. In 1979 he joined the Philips Research Laboratories in Eindhoven. He led research projects in image processing, and participated in European projects. He has coached students from various universities, and since 1988 has taught at the Philips Centre for Technical Training. In 1991/1992, he was a visiting researcher in the Information Theory Group of Delft University. In 1994, he was a guest editor for Signal Processing: "Image Communications" for a special issue on Video Format Conversion. At present, he is a Senior Scientist in the Television Systems group of Philips Research and has a particular interest in algorithms for motion estimation, scan-rate conversion, and image enhancement. His work in these areas has resulted in about 35 patents and patent applications. He is a Senior Member of the IEEE, and received the first prize in the 1995 ICCE Outstanding Paper Awards program and the second prize in 1998. The Philips 100 Hz TV with 'Natural Motion', based on his PhD study, received the European Innovation Award of the Year 95/96 from the European Imaging and Sound Association.